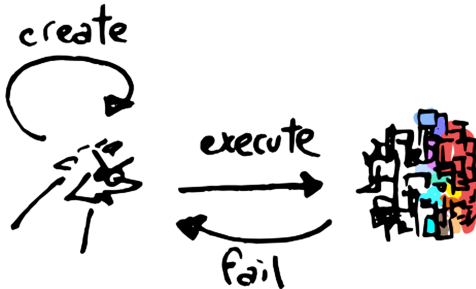


# Verification of infinite state systems

Gabriele Puppis

LaBRI / CNRS



## Outline of the course

### ① **Warm-up**

(transition systems, automata, logics)

### ② **First-order theories**

(undecidability, Presburger logic, automatic structures)

### ③ **The monadic theory of one successor**

(contraction and composition methods, factorization forests)

### ④ **The monadic theory of two successors**

(Rabin's complementation, application examples)

### ⑤ **The transformational approach**

(interpretations, context-free and prefix-rewriting graphs, unfoldings, Caucal hierarchy, recursive program schemes)

### ⑥ **Reachability via saturation**

(pushdown systems, VAS / Petri nets, lossy counter machines)

## Goal

Automatic verification of **properties** of **systems**.

## Goal

Automatic verification of **properties** of **systems**.

- which properties?

safety	<i>"something bad never happens"</i>	} reachability
liveness	<i>"something good eventually happens"</i>	
fairness	<i>"if something happens infinitely often then something else eventually happens"</i>	
formulas	<i>"<math>\forall t. \exists t'. t \leq t' \wedge a(t')</math>"</i>	

## Goal

Automatic verification of **properties** of **systems**.

- which properties?

safety	<i>"something bad never happens"</i>	} reachability
liveness	<i>"something good eventually happens"</i>	
fairness	<i>"if something happens infinitely often then something else eventually happens"</i>	
formulas	<i>"<math>\forall t. \exists t'. t \leq t' \wedge a(t')</math>"</i>	

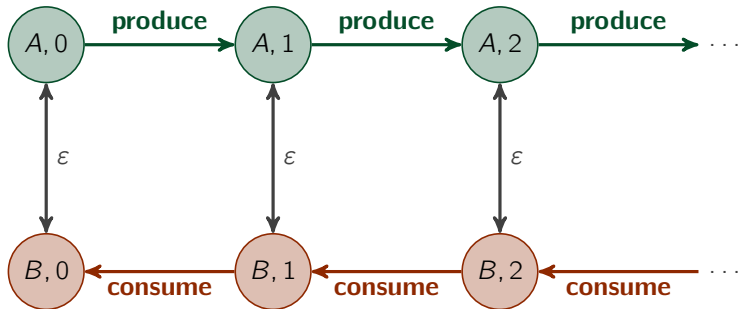
- which systems?

reactive	<i>"transitions enabled on the basis of input"</i>
infinite	{ <i>stacks (or recursion)</i> <i>variables</i> <i>queues</i> <i>lists</i>

A: repeat forever  
do atomically  
produce  
 $count := count + 1$



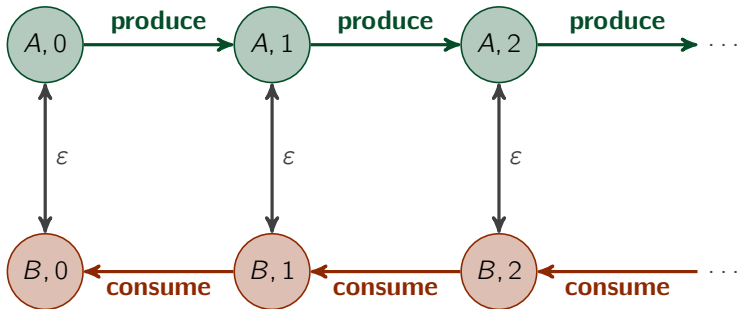
B: repeat forever  
do atomically  
if  $count > 0$  then  
consume  
 $count := count - 1$

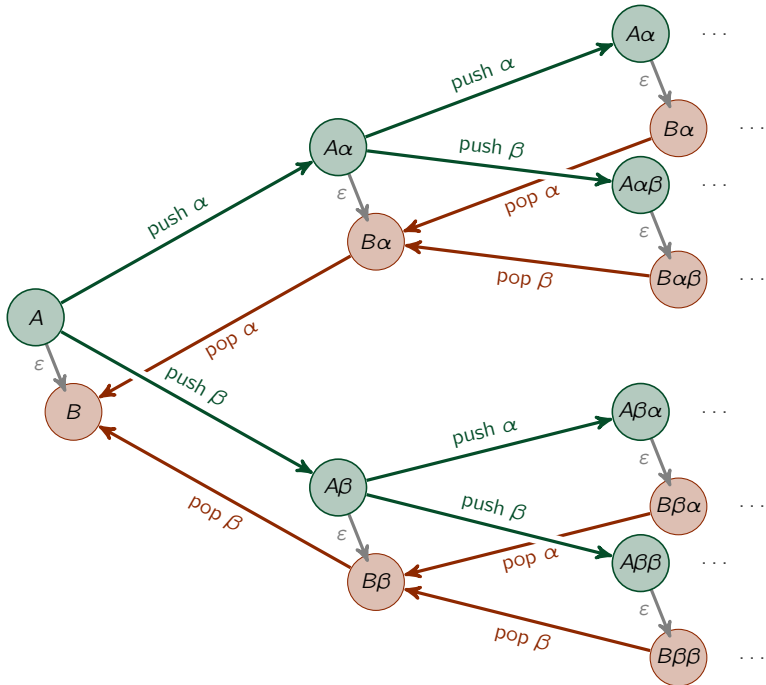


## Definition

A **transition system** is a graph  $G = ((V_a)_{a \in \Sigma}, (E_b)_{b \in \Delta})$  where

- vertices are associated with labels from a finite alphabet  $\Sigma$
- edges are associated with labels from a finite alphabet  $\Delta$







Automata = finite transition systems

👉 but mostly used as representations of **languages**

Automata = finite transition systems

👉 but mostly used as representations of **languages**

### Definition

A **finite state automaton** is a tuple  $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ , where

- $Q$  is a finite set of control states
- $\Sigma$  is a finite alphabet for transition labels
- $\Delta \subseteq Q \times \Sigma \times Q$  is a finite set of transition rules
- $I \subseteq Q$  is a set of initial states
- $F \subseteq Q$  is a set of final states

Automata = finite transition systems

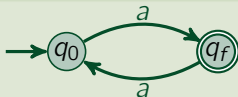
👉 but mostly used as representations of **languages**

## Definition

A **finite state automaton** is a tuple  $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ , where

- $Q$  is a finite set of control states
- $\Sigma$  is a finite alphabet for transition labels
- $\Delta \subseteq Q \times \Sigma \times Q$  is a finite set of transition rules
- $I \subseteq Q$  is a set of initial states
- $F \subseteq Q$  is a set of final states

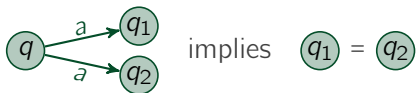
## Example



$$\mathcal{L}(\mathcal{A}) = a (a a)^*$$

Different types of automata:

- **deterministic**



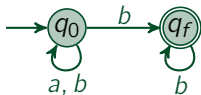
- **with  $\varepsilon$ -transitions**



- **complete**

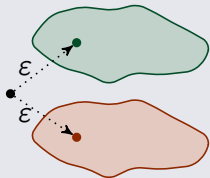


- **Büchi/parity conditions**

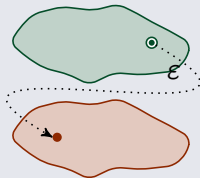


# Language theoretic operations on automata

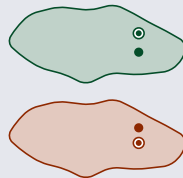
**union**



**concatenation**

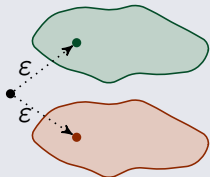


**complementation**

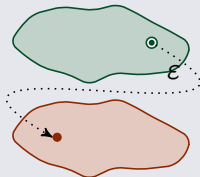


# Language theoretic operations on automata

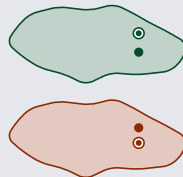
**union**



**concatenation**

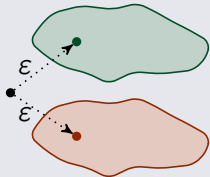


**complementation**

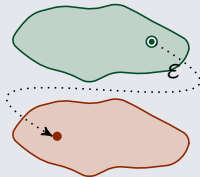


# Language theoretic operations on automata

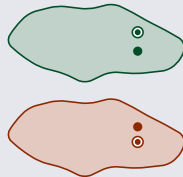
## union



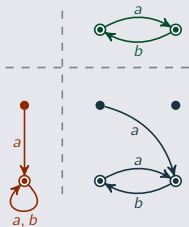
## concatenation



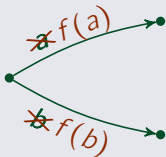
## complementation



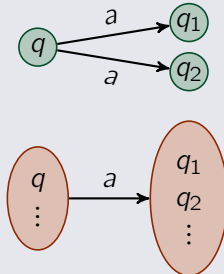
## intersection



## projection



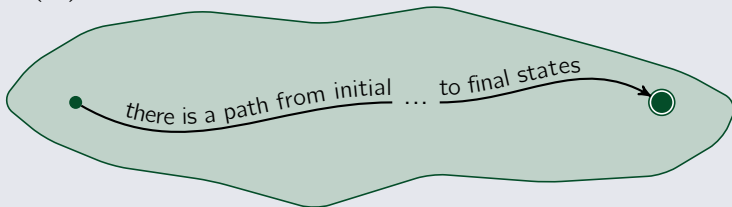
## subset construction



## Problems on automata

- **Non-emptiness**

$$\mathcal{L}(\mathcal{A}) \neq \emptyset \text{ iff}$$



- **Universality**

$$\mathcal{L}(\mathcal{A}) = \Sigma^* \text{ iff } \mathcal{L}(\mathcal{A}^c) = \emptyset$$

- **Containment**

$$\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B}) \text{ iff } \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B}^c) = \emptyset$$



These are simple graph search problems!



## Logics for specification of properties

- **Propositional logic**



$$b \vee \neg b$$

## Logics for specification of properties

- **Propositional logic**



$$b \vee \neg b$$

- **First-order logic**

$$a(x_0) \wedge \forall x. (a(x) \rightarrow b(x)) \rightarrow b(x_0)$$



## Logics for specification of properties

- **Propositional logic**



$$b \vee \neg b$$

- **First-order logic**

$$a(x_0) \wedge \forall x. (a(x) \rightarrow b(x)) \rightarrow b(x_0)$$



- **Monadic second-order logic**

$$\exists Z. \forall x. \exists y. (Z(y) \wedge y = x + 1)$$

## Examples of sentences and formulas

$$\psi_{\text{dense}} = \forall x, y. \exists z. (x < y \rightarrow x < z \wedge z < y)$$

$$\psi_{\text{connected}} = \forall Z. \left( \exists x, y. Z(x) \wedge \neg Z(y) \right) \rightarrow \\ \left( \exists x, y. Z(x) \wedge \neg Z(y) \wedge E(x, y) \right)$$

$$\psi_{\text{path}}(x, y) = \forall Z. Z(x) \wedge \\ \forall x', y'. (Z(x') \wedge E(x', y') \rightarrow Z(y')) \rightarrow Z(y)$$

## Examples of sentences and formulas

$$\psi_{\text{dense}} = \underbrace{\forall x, y. \exists z. (x < y \rightarrow x < z \wedge z < y)}_{\text{FO}[<] \text{ over } \mathbb{Q}}$$

$$\psi_{\text{connected}} = \underbrace{\forall Z. \left( \exists x, y. Z(x) \wedge \neg Z(y) \right) \rightarrow \left( \exists x, y. Z(x) \wedge \neg Z(y) \wedge E(x, y) \right)}_{\text{MSO}[E] \text{ over } G=(V, E)}$$

$$\psi_{\text{path}}(x, y) = \forall Z. Z(x) \wedge \underbrace{\forall x', y'. \left( Z(x') \wedge E(x', y') \rightarrow Z(y') \right) \rightarrow Z(y)}_{\text{MSO}[E] \text{ over } G=(V, E)}$$

 The underlying **signature** and **domain** are important!

- $\text{MSO}[+1] \text{ over } \mathbb{N} = \text{MSO}[<] \text{ over } \mathbb{N}$
- $\text{FO}[0, 1, +] = \text{Presburger arithmetic}$
- $\text{MSO over } \mathbb{N} = \text{FO}[\subseteq] \text{ over } 2^{\mathbb{N}}$
- $\text{FO}[\epsilon] \text{ over models of Zermelo–Fraenkel set theory...}$

## Other examples of MSO properties

$$\begin{aligned}\psi_{3\text{-colorability}} = & \exists X, Y, Z. \forall v. (X(v) \vee Y(v) \vee Z(v)) \\ & \forall u, v. E(u, v) \rightarrow \neg(X(u) \wedge X(v)) \wedge \\ & \neg(Y(u) \wedge Y(v)) \wedge \\ & \neg(Z(u) \wedge Z(v))\end{aligned}$$

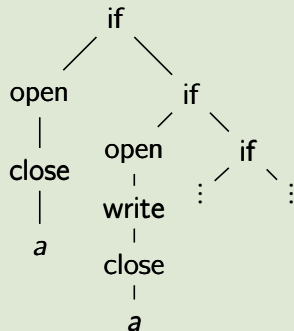
$$\psi_{K5}(x_1, \dots, x_5) = \bigwedge_{i \neq j} (x_i \neq x_j \wedge \psi_{\text{path}}(x_i, x_j))$$

$$\psi_{K3,3}(x_1, \dots, x_3, y_1, \dots, y_3) = \bigwedge_{i \neq j} (x_i \neq x_j \wedge y_i \neq y_j) \wedge \bigwedge_{i,j} \psi_{\text{path}}(x_i, y_j)$$

$$\begin{aligned}\psi_{\text{planar}} = & \neg \exists x_1, \dots, x_5. \psi_{K5}(x_1, \dots, x_5) \wedge \\ & \neg \exists x_1, \dots, x_3, y_1, \dots, y_3. \psi_{K3,3}(x_1, \dots, x_3, y_1, \dots, y_3)\end{aligned}$$

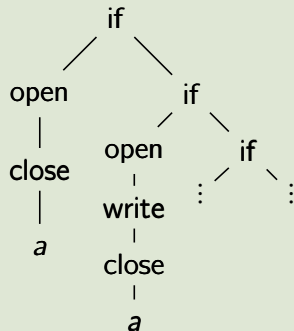
## A real example

```
let  $Foo(g, h) =$   
   $g(h)$   
  if [user hits key] then  
     $g \cdot close(h)$   
  else  
     $Foo(g \cdot write, h)$   
 $Foo(open, a)$ 
```



## A real example

```
let  $Foo(g, h) =$   
   $g(h)$   
  if [user hits key] then  
     $g \cdot close(h)$   
  else  
     $Foo(g \cdot write, h)$   
 $Foo(open, a)$ 
```



One may want to verify that all sequences of write operations occur between open and close operations:

$$\forall Z \text{ path. } \forall z \in Z. \text{ write}(z) \rightarrow \exists x, y \in Z. x < z < y \wedge \text{open}(x) \wedge \text{close}(y)$$



▶ Next